Constriction Computation using Surface Curvature

Franck Hétroy

Constriction Computation using Surface Curvature

F. Hétroy

GRAVIR - IMAG/INRIA, Grenoble, France

Abstract

This paper provides a curvature-based algorithm to compute locally shortest geodesics on closed triangulated surfaces. These curves, which are called "constrictions", are useful for shape segmentation. The key idea of the algorithm is that constrictions are almost plane curves; it first finds well-located simple, plane, closed curves, and then slides them along the surface until a shortest geodesic is reached. An initial curve is defined as a connected component of the intersection between the surface and a plane going through an initial vertex. Initial vertices and planes are determined using approximations of surface curvature.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction and related work

Automatic detection of particular, well-defined, object features can be of great interest for many applications. For example, detection of the narrow parts of an object defined by its boundary surface (usually represented as a triangulated mesh) has been applied to solve at least three different problems. In [DGG03], such parts are defined as subsets of the object volume, and called "stable manifolds". They are used for shape segmentation and, subsequently, shape matching. Another approach is to define these parts as simple, closed curves on the surface. Such curves correspond to bottlenecks of the object if they locally minimize length (that is to say, if every other simple, closed curve in a tubular neighborhood has greater length). Figure 1 gives an example of such a curve.

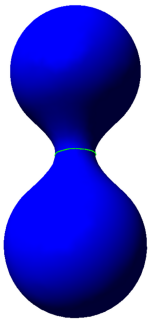

Figure 1: *Example of a constriction.*

This approach has been developed in [HA03a], in which these curves are called "constrictions". They are closed geodesics, and the angle of a constriction at each of the vertices it goes through is at least π. Such constrictions can be used to segment shapes; but computing constrictions can also be useful in order to detect medical diseases, such as so called airway or vascular constrictions [BWK05]. As far as we are concerned, we plan to apply constriction detection to character animation in a future work.

The algorithm given in [HA03a] to find simple, closed curves near constrictions, and then to slide them to reach constrictions [HA03b], has several drawbacks. First, its computation time seems prohibitive, since it requires progressive surface simplification. Second, it is unable to compute some special configurations of constrictions, such as orthogonal constrictions (see Fig. 4). This is because finding a seed curve during the surface simplification stage sometimes prevents to find other neighboring seed curves, since surface cannot be simplified anymore around these curves.

In this paper, we propose a new algorithm to detect constrictions on closed triangulated surfaces which overcomes these problems. This algorithm first finds initial simple, closed curves on the surface, and then slides them to reach constrictions, using the algorithm of [HA03b]. Detection of initial curves must satisfy two major requirements:

1. the number of curves computed must be relevant for the application;
2. these curves must be located as close as possible to constrictions, in order to reduce the computation time of the sliding algorithm, and not to slide to a single vertex (degenerated constriction).

Our algorithm first chooses some special vertices on the sur-

face, and then computes initial plane curves starting from these vertices. Selection of these vertices and of the underlying planes of the curves is based on surface curvature, and is detailed on section 3. We have chosen to make use of surface curvature because we believe there is a strong link between constrictions and surface curvature; in particular constrictions often follow principal curvature directions.
The following section recalls some definitions and properties about constrictions and curvatures. Section 3 then describes our algorithm. Results are shown and discussed in section 4, and we conclude in last section.

2. Constrictions and curvature

2.1. Constrictions on closed surfaces

Let $\mathcal{P}$ be a 2-manifold embedded in $\mathbb{R}^3$. Let α be a simple, closed curve on $\mathcal{P}$. α is a *constriction* if there exists $\varepsilon > 0$ such that every simple, closed curve on $\mathcal{P}$ whose Hausdorff distance to α is lower than ε has greater length than α's [HA03a]. It follows that constrictions are closed geodesics on $\mathcal{P}$.
We now only consider the case of polyhedral surfaces. A vertex through which a curve goes is called a *pivot vertex*. The *angle made by a curve* α *at a pivot vertex* v is the minimum of the two angles made by α on the surface around v [HA03a]. If α is a constriction, then the angle made by α at each of its pivot vertices is at least π. As a consequence, for each pivot vertex v of a constriction, the angle of the surface at v (that is to say, the sum of the angles of the triangles which are adjacent to v at v) is at least 2π. Such vertices are called *saddle vertices*. Constrictions can be characterized as closed geodesics with angle at least π at each of their pivot vertices [HA03a].

2.2. Geodesics and surface curvature

Constrictions seem to be closely related to surface curvature. In particular, experience shows that constrictions are special lines of curvature. The algorithm described in next section thus constructs constrictions on closed, triangulated surfaces using surface curvature. But instead of computing principal curvature directions on the whole surface, which can be time prohibitive, it uses the following property for smooth surfaces, which can be for example found as an exercise in [dC76], p. 260:

- if a curve on a smooth surface is both a line of curvature and a geodesic, then it is a plane curve;
- if a (nonrectilinear) geodesic on a smooth surface is a plane curve, then it is a line of curvature.

Since, in computer graphics, polyhedral surfaces often approximate smooth surfaces, we hope to find constrictions as approximations of plane curves.

3. Algorithm

Our algorithm relies on the idea that a constriction on a polyhedral surface approximates a plane curve. Thus, if we find both a vertex through which the constriction goes and the corresponding plane, we can theoretically construct a constriction. In fact, since this is just an approximation, we need to slightly move the curve to find a constriction. Our algorithm proceeds in three successive stages:

1. we find candidate vertices to construct the curves;
2. we compute the corresponding planes and the intersections between these planes and the surface, which give us the initial curves;
3. we slide these curves to reach constrictions, using the algorithm of [HA03b].

Note that, theoretically, a very huge number of constrictions can exist on a polyhedral surface, including constrictions very close to others. The goal of this algorithm is to compute only a few relevant ones.

3.1. Choice of initial vertices

The first stage of the algorithm is to find vertices (hopefully) belonging to constrictions. We can think of interactive selection of vertices [BWK05]; however it can take a long time, and if the shape of the surface is complex (this can be the case in some medical applications, for example), some constrictions may be forgotten. That is why we prefer an automatic solution, even though some parameters must be selected by hand. Since each pivot vertex of a constriction is a saddle vertex, the simplest idea is to choose, as initial vertices, the saddle vertices of the surface. Unfortunately, on a standard triangulated surface, about half of the vertices are saddle vertices, which is far too much for our purpose. Another idea can be to choose only vertices on which the surface angle is locally maximum. But such vertices also are too numerous. Actually, the problem is that we only need *one* vertex to construct a constriction, and not all vertices through which it goes.
Since constrictions correspond to concave areas on the surface, we decided to select initial vertices according to the (oriented) angles made between successive adjacent faces around a vertex: if the sum of these angles is lower than a user-defined (negative) threshold, the vertex is selected as an initial vertex. More formally, let v be a vertex on the surface, and let $f_1, f_2, \ldots, f_n, f_{n+1} = f_1$ be the successive faces which are adjacent to v. Let $\theta_i, 1 \leq i \leq n$, be the oriented angle between f_i and f_{i+1}. θ_i is positive if the oriented edge e_i incident to f_i and f_{i+1} is convex, and negative if e_i is concave. Then v is selected as initial vertex if, and only if, $\sum_{i=1}^{n} \theta_i < T$, where T is a user-defined threshold.
θ_i is sometimes used as a coarse approximation of the mean curvature of the surface around e_i. For sake of simplicity and in order to speed up the algorithm, we did not choose to compute other more precise approximations of mean curvature, such as those defined in [CP03, CSM03], although it would be interesting to compare results.

3.2. Computation of initial curves

An initial curve c is defined as a connected component of the intersection between the surface and a plane p to which an initial vertex v belongs. In order to define a plane, since we know a point on this plane (the initial vertex v), we need to choose two directions. These directions are computed using principal curvature directions, as follows:

- first, maximum (or minimum) principal curvature direction at v is computed, using the algorithm described in [CSM03, ACSD*03].
- Then, this direction is projected on a plane p' to which v belongs. p' is defined by its normal, which is the average sum of the normals of the incident faces f_i, with respect to the angles made by the faces at v. This normal can be seen as an approximation of the normal of the surface at v.
- Each face f_i is also projected on p'. The projected face to which the projected maximum curvature direction belongs is selected, and the maximum curvature direction is re-projected on the corresponding face, to a segment vw which gives us the first of our two directions. This method ensures that we get one, and only one, projection of the direction on the surface.
- Maximum (or minimum) principal curvature direction at w is computed, as the weighted average between maximum (or minimum) principal curvatures at the two endpoints of the edge to which w belongs.
- This direction is also projected onto the surface, in order to get our second direction. This case is simpler than the previous one, since we know to which face of the surface the projected direction belongs: it is the face which is incident to the edge to which w belongs, but not adjacent to v.

Instead of computing a second maximum principal curvature direction in order to define the plane p, another idea can be to choose the minimum principal curvature direction at v (which is orthogonal to the maximum one) as the normal of p. Since p is defined using only one vertex, this possibility gives results very sensitive to the accuracy of the curvature computation.
We have chosen to compute principal curvature directions with the algorithm of Cohen-Steiner et al. because it gives fast and accurate results. Accurate approximation of principal curvature directions is critical in order to get initial curves not too far from corresponding constrictions. Also, note that we do not compute curvature directions on every vertex, but only on initial vertices. This speeds up the algorithm.
The initial curve going through v is then computed as the connected component of the intersection between the surface and the plane, to which v belongs (actually, the curve is computed starting from v).
Once the initial curves have been computed, we slide them to exact location of neighboring constrictions using the algorithm described in [HA03b].

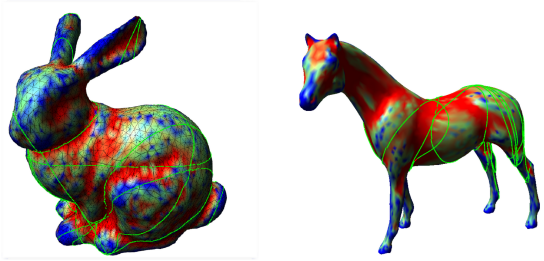

Figure 2: *Approximation of mean curvature and starting curves computed on two standard models.*

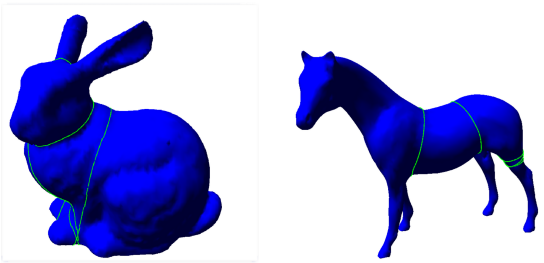

Figure 3: *Constrictions computed on two standard models.*

4. Results and discussion

We have tested our algorithm on several closed, triangulated surfaces. Figure 2 shows our mean curvature approximation computed on a bunny and a horse model (each model having 10,000 faces), together with the initial curves detected. "Hot" areas correspond to vertices with the lowest mean curvature, that is to say the vertices we are interested in as initial vertices. Of course, these areas correspond to concave areas. The threshold we used was $-\pi/9$ (i.e., -20^o) for the bunny and $-\pi/18$ for the horse.
We can observe that some initial curves seem to be close to constrictions, while other (the longest ones) seem not. This is mainly because the longest the curve, the more important the accuracy of principal curvature direction: in these cases, a small error in the direction leads to a curve which can be located far away from a constriction.
Figure 3 shows the constrictions computed on both of these models. Several initial curves are slided to a degenerated constriction, i.e. to a single point, while several others are slided to the same constriction: the number of initial curves is greater than the number of computed constrictions, as in [HA03b]. Several neighboring constrictions are detected on the back left horse leg. This is because the surface is noisy in this area, mainly because the model has a low density of vertices there. What is important to point out is that this method is able to construct "complicated" constrictions, for example constrictions wrapping around several parts of a model (as the longest constriction of the bunny model does), while method described in [HA03a] cannot. Moreover, orthogonal

Model	T	Init. curves	Constrictions
Star	$-\pi/36$	19	2
Star	$-\pi/18$	19	2
Star	$-\pi/9$	0	0
Noisy star	$-\pi/36$	39	4
Noisy star	$-\pi/18$	10	3
Noisy star	$-\pi/9$	1	1

Table 1: *Number of initial curves and constrictions computed on a star model, using different threshold values.*

Model	Nb. faces	Threshold	Init. curves	Time
Noisy star	5,120	$-\pi/18$	10	25 sec
2 crescents	4,320	$-\pi/18$	7	10 sec
Bunny	10,000	$-\pi/9$	14	80 sec
Horse	10,000	$-\pi/18$	16	27 sec
Cat	5,000	$-\pi/18$	31	60 sec
Dino	10,000	$-\pi/18$	15	60 sec
Dragon	10,000	$-\pi/9$	45	42 sec

Table 2: *Computation times.*

constrictions can also be computed, as can be seen on Fig. 4.

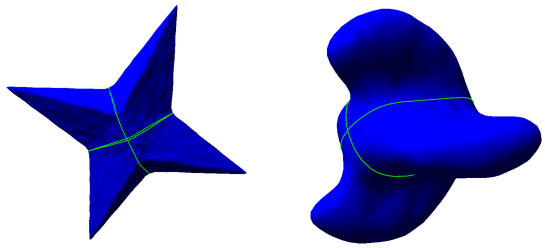

Figure 4: *Contrary to [HA03a], orthogonal constrictions can be computed with our algorithm.*

The choice of the threshold T is important: the lowest the threshold (in absolute value), the greatest the number of initial vertices, thus the greatest the number of constrictions. For example, Table 1 gives the number of initial curves and constrictions computed on a noisy or not star model (Fig. 4 left; several initial curves slide to the same constriction). If T is too low, irrelevant constrictions can be computed, slowing down the algorithm. But if T is too high, some relevant constrictions can be missed (e.g. left ear of the bunny and ankles of the horse on Fig. 3).

Computation times for several models are given in Table 2. Tests were performed using a Pentium IV 2.4 GHz with 512 MB RAM. Note that code was not optimized. Differences between computation times for models with the same number of faces is due both to the number of initial curves and to their distance to a constriction. We think these times can be significantly improved if the choice of initial vertices and the position of corresponding planes are improved, because computing a geodesic is the slowest task; thus the less geodesics are to be computed using [HA03b], the fastest the algorithm will be.

5. Conclusion and future work

In this paper, we have proposed a new, simple algorithm to compute constrictions on closed, triangulated surfaces. The algorithm computes initial plane curves starting from selected vertices, and slides them to reach constrictions. Initial vertices are selected according to the sum of the angles of adjacent faces around them; planes are constructed starting from these vertices and projecting maximum principal curvature directions on the surface. First results are encouraging, but this algorithm can still be improved. For example, we can think of other choice of initial vertices. Another solution could be to create an ordered list of candidate vertices, according to some criterion (for example, the value of our approximation of mean curvature), which are proposed to the user. To speed up the computation of constrictions, more recent and fast geodesic computation algorithms, such as [SSK*05], may be used. From a theoretical point of view, it would be interesting to deeply investigate relationship between constrictions and surface curvature. Finally, we plan to apply these results to character animation (constrictions will be used to detect articulation areas). In this case, we need to compute only some special constrictions, and not all of them. Hence further assumptions have to be made.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LEVY B., DESBRUN M.: Anisotropic polygonal remeshing. In *SIGGRAPH* (2003), pp. 485–493.

[BWK05] BISCHOFF S., WEYAND T., KOBBELT L.: Snakes on triangle meshes. In *Bildverarbeitung für die Medizin* (2005), pp. 208–212.

[CP03] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. In *Symp. on Geometry Processing* (2003), pp. 177–187.

[CSM03] COHEN-STEINER D., MORVAN J.: Restricted delaunay triangulations and normal cycle. In *ACM Symp. on Computational Geometry* (2003), pp. 237–246.

[dC76] DO CARMO M.: *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[DGG03] DEY T., GIESEN J., GOSWAMI S.: Shape segmentation and matching with flow discretization. In *Workshop on Algo. and Data Struct.* (2003), pp. 25–36.

[HA03a] HÉTROY F., ATTALI D.: Detection of constrictions on closed polyhedral surfaces. *Data Visualization* (2003), 67–74. Proc. Visualization Symp.

[HA03b] HÉTROY F., ATTALI D.: From a closed piecewise geodesic to a constriction on a closed polyhedral surface. In *Pacific Graphics* (2003), pp. 394–398.

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S., HOPPE H.: Fast exact and approximate geodesics on meshes. In *SIGGRAPH* (2005).